EEDG/CE 6303: Testing and Testable Design (Spring'2023)

Department of Electrical & Computer Engineering

The University of Texas at Dallas

Instructor: Mehrdad Nourani (nourani@utdallas.edu)

Cover Page for All Submissions

(Assignment, Project, Codes/Simulations/CAD, Examinations, etc.)

Last Name (as shown in the official UT Dallas Student ID Card): Annadata

First Name: Yagna Srinivasa Harsha

Submission Materials for (e.g. Homework #, Project #): H o m e w o r k 4

Statement of Academic Honesty

I certify that:

i. the attached report (for assignment, project, codes/simulations/CAD, examinations, etc.) is my own work, based on my personal study and/or research,
ii. I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication,
iii. this report has not previously been submitted for assessment in EEDG/CE 6303 or any other course at UT Dallas or elsewhere,
iv. I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons, and
v. I have read and understood the Department and University policies on scholastic dishonesty as outlined in: http://www.utdallas.edu/deanofstudents/dishonesty/.

Name: Y a g n a S H A n n a d a t a Date: 4 / 1 3 / 2 0 2 3

Signature: 

Yagna Srinivasa Harsha Annadata
CE 6303
Yxa210024

Assignment – 4

(Q2)

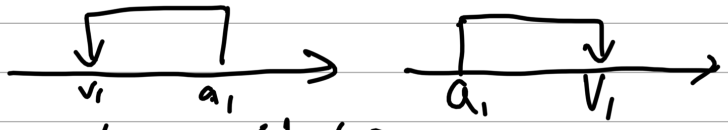
$$\{M0 :\Updownarrow (w0); M1 :\Uparrow (r0, w1); M2 :\Downarrow (r1, w0); M3 :\Updownarrow (r0)\}$$

March X: AF's, SAF's, TF's

fault	condition	Sensitization	detection	Comments.
AF	N/A	N/A	N/A	$M_1 + M_2$
SAF <↑/0>	N/A	M1 → tries to write '1'	M2 → reads '1' and expects '1'	N/A
SAF <↑/1>	N/A	M0 → tries to write '0'	M1 → reads '0' and expects '0'	M3 also detects.
TF <↑/0>	N/A	M0	M1	N/A
TF <↓/1>	N/A	M1	M2	N/A

(iv) Unlinked CF's

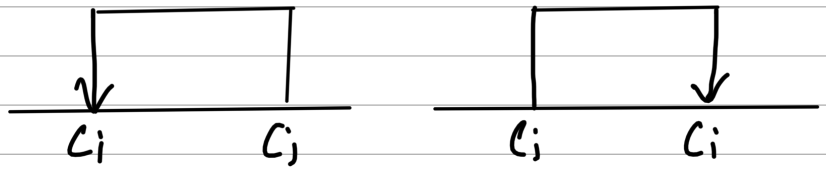
(a) CFid.



$\langle\uparrow,0\rangle, \langle\uparrow/1\rangle, \langle\downarrow/0\rangle, \langle\downarrow/1\rangle$

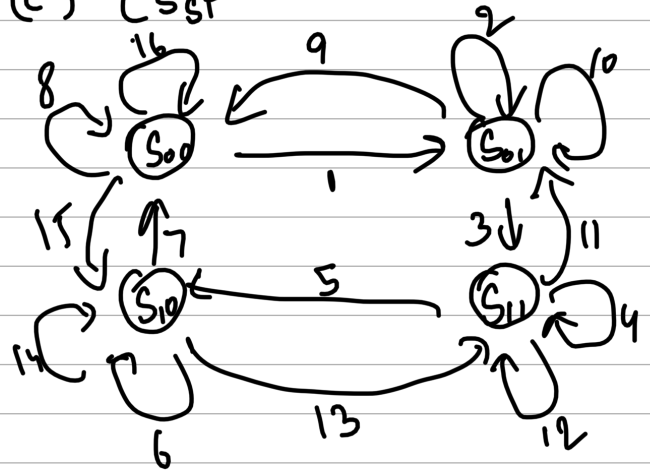
fault	condition	sensitization	detection	comments
$\langle\uparrow/0\rangle$	a-cell < V-cell	N/A	N/A	N/A
$\langle\uparrow/1\rangle$	a-cell < V-cell	M1	M2	N/A
$\langle\downarrow/0\rangle$	a-cell < V-cell	N/A	N/A	N/A
$\langle\downarrow/1\rangle$	a-cell < V-cell	M2	M3	N/A
$\langle\uparrow/0\rangle$	V-cell < a-cell	M1	M2	N/A
$\langle\uparrow/1\rangle$	V-cell < a-cell	N/A	N/A	N/A
$\langle\downarrow/0\rangle$	V-cell < a-cell	M1	M2	N/A
$\langle\downarrow/1\rangle$	V-cell < a-cell	N/A	N/A	N/A

(b) CFin



fault	condition	sensitization	detection	comments
$\langle\uparrow/\updownarrow\rangle$	j < i	M1	M1	N/A
$\langle\downarrow/\updownarrow\rangle$	j < i	M2	M3	N/A
$\langle\uparrow/\updownarrow\rangle$	i < j	M1	M2	N/A
$\langle\downarrow/\updownarrow\rangle$	i < j	M2	M2	N/A

(c) CSst



i < j

step	march	State	Operation	State
1	M0	S00	w0 i=0	S00
2	M0	S00	w0 j=0	S00
3	M1	S00	r0 i=0	S00
4	M1	S00	w1 i=1	S01
5	M1	S01	r0 j=0	S01
6	M1	S01	w1 i=1	S11
7	M2	S11	r1 i=1	S11
8	M2	S11	w0 i=0	S10
9	M2	S10	r0 j=0	S10
10	M2	S10	w0 j=0	S00
11	M3	S00	r0 i=0	S00
12	M3	S00	r0 j=0	S00

j < i

step	march	State	Operation	State
1	M0	S00	w0 j=0	S00
2	M0	S00	w0 i=0	S00
3	M1	S00	r0 j=0	S00
4	M1	S00	w1 j=1	S10
5	M1	S10	r0 i=0	S10
6	M1	S10	w1 i=1	S11
7	M2	S11	r1 j=1	S11
8	M2	S11	w0 j=0	S01
9	M2	S01	r0 i=0	S01
10	M2	S01	w0 i=0	S00
11	M3	S00	r0 j=0	S00
12	M3	S00	r0 i=0	S00

(V) linked CF's

consider $a_1 \rightarrow \langle \uparrow / 0 \rangle$
$a_2 \rightarrow \langle \uparrow / 1 \rangle$

fault	condition	Sensitizing	Detection	Conceal
$\langle \uparrow / 0 \rangle$ a_1 $\langle \uparrow / 1 \rangle$ a_2	a_2 - cell < a_1 cell < V cell	M_1 $a_2 \rightarrow 0 > 1$ V → 1 $a_1 \rightarrow 0 > 1$ V → 0	N/A	N/A

Conclusion:

March x test can detect AF, SAF, TF Unlinked CFs, but no linked CF.

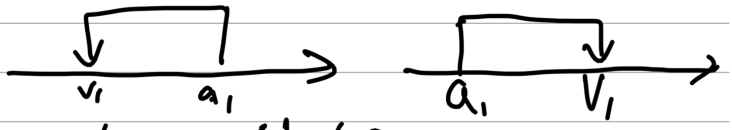
(Q3) $\{M0 : \Updownarrow (w0); M1 : \Uparrow (r0, w1, r1); M2 : \Downarrow (r1, w0, r0); M3 : \Updownarrow (r0)\}$

March Y: AF's, SAF's, TF's

fault	Condition	Sensitization	detection	Comments.
AF	N/A	N/A	N/A	$M_1 + M_2$
SAF $\langle \forall / 0 \rangle$	N/A	M1 → tries to write '1'	M1 → reads '1' and expects '1'	M2 also detects.
SAF $\langle \forall / 1 \rangle$	N/A	M2 → tries to write '0'	M2 → reads '0' and expects '0'	M3 also detects.
TF $\langle \uparrow / 0 \rangle$	N/A	M0	M1	N/A
TF $\langle \downarrow / 1 \rangle$	N/A	M1	M2	N/A

(iv) Unlinked CFs

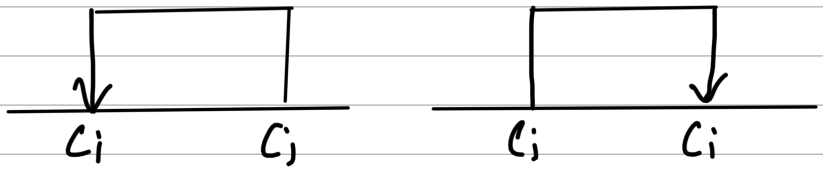
(a) CFid.



$\langle\uparrow/0\rangle, \langle\uparrow/1\rangle, \langle\downarrow/0\rangle, \langle\downarrow/1\rangle$

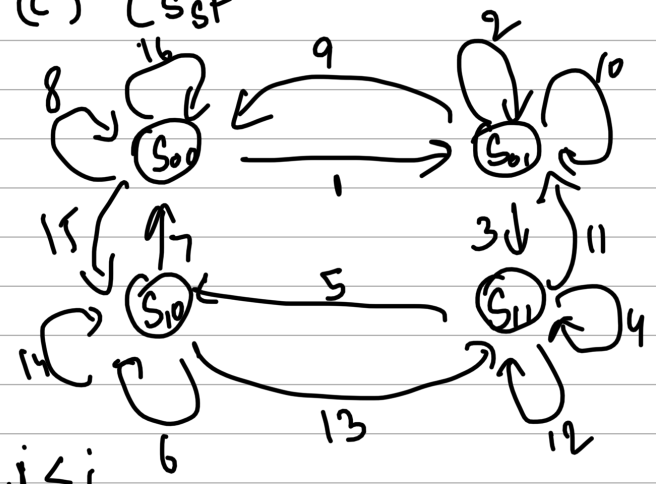
fault	condition	sensitization	detection	comments.
$\langle\uparrow/0\rangle$	a-cell < V-cell	N/A	N/A	N/A
$\langle\uparrow/1\rangle$	a-cell < V-cell	M_1 → read '0' write '1' to a-cell	M_1 → reads '0' But V-cell = '1'	N/A
$\langle\downarrow/0\rangle$	a-cell < V-cell	N/A	N/A	N/A
$\langle\downarrow/1\rangle$	a-cell < V-Sell	M2	M2	N/A
$\langle\uparrow/0\rangle$	V-cell < a-cell	M1	M1	N/A
$\langle\uparrow/1\rangle$	V-cell < a-cell	N/A	N/A	N/A
$\langle\downarrow/0\rangle$	V-cell < a-cell	M1	M1	N/A
$\langle\downarrow/1\rangle$	V-cell < a-cell	N/A	N/A	N/A

(b) CFin



fault	condition	sensitization	detection	comments
$\langle\uparrow/\updownarrow\rangle$	j < i	M1	M1	N/A
$\langle\downarrow/\updownarrow\rangle$	j < i	M2	M2	N/A
$\langle\uparrow/\updownarrow\rangle$	i < j	M1	M1	N/A
$\langle\downarrow/\updownarrow\rangle$	i < j	M2	M2	N/A

(c) CSst



j < i

step	mach	State	Operation	Stati
1	M0	S_{00}	w0 i=0	S_{00}
2	M0	S_{00}	w0 j=0	S_{00}
3	M1	S_{00}	r0 j=0	S_{00}
4	M1	S_{00}	wi j=1	S_{10}
5	M1	S_{10}	ri j=1	S_{10}
6	M1	S_{10}	r0 i=0	S_{10}
7	M1	S_{10}	w1 i=1	S_{11}
8	M1	S_{11}	ri i=1	S_{11}
9	M2	S_{11}	ri j=1	S_{11}
10	M2	S_{11}	w0 j=0	S_{01}
11	M2	S_{01}	r0 j=0	S_{01}
12	M2	S_{01}	ri i=1	S_{01}

i < j

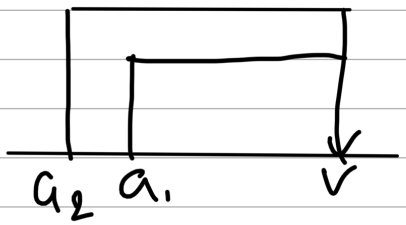
step	mach	State	Operation	Stati
1	M0	S_{00}	w0 i=0	S_{00}
2	M0	S_{00}	w0 j=0	S_{00}
3	M1	S_{00}	r0 i=0	S_{00}
4	M1	S_{00}	wi i=1	S_{01}
5	M1	S_{01}	ri i=1	S_{01}
6	M1	S_{01}	r0 j=0	S_{01}
7	M1	S_{01}	w1 j=1	S_{11}
8	M1	S_{11}	ri j=1	S_{11}
9	M2	S_{11}	ri i=1	S_{11}
10	M2	S_{11}	w0 i=0	S_{10}
11	M2	S_{10}	r0 i=0	S_{10}
12	M2	S_{10}	ri j=1	S_{10}
13	M2	S_{10}	w0 j=0	S_{00}
14	M2	S_{00}	r0 j=0	S_{00}
15	M3	S_{00}	r0 i=0	S_{00}
16	M3	S_{00}	r0 j=0	S_{00}

j < i

13	M2	S_{01}	w0 i=0	S_{00}
14	M2	S_{00}	r0 i=0	S_{00}
15	M3	S_{00}	r0 i=0	S_{00}
16	M3	S_{00}	r0 j=0	S_{00}

(V) linked CF's

consider $a_1 \rightarrow \langle \uparrow / 0 \rangle$
$a_2 \rightarrow \langle \uparrow / 1 \rangle$



fault	condition	sensitizing	Detection	Conceal
$\langle \uparrow / 0 \rangle$ a_1 $\langle \uparrow / 1 \rangle$ a_2	a_2 - cell $< a_1$ cell $< V$ cell	M_1 $a_2 \rightarrow 0 > 1$ $V \rightarrow 1$ $a_1 \rightarrow 0 > 1$ $V \rightarrow 0$	N/A	N/A

Conclusion:

March Y can detect AF, SAF, TF, Unlinked CF but cannot detect linked CF.

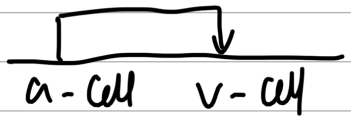
(Q1)

(i) Let us consider new fault model

Notation: $\langle w0/\uparrow \rangle, \langle w0/\downarrow \rangle, \langle w1/\uparrow \rangle, \langle w1/\downarrow \rangle, \langle r0/\uparrow \rangle, \langle r0/\downarrow \rangle$
$\langle r1/\uparrow \rangle, \langle r1/\downarrow \rangle$

Consider the following MARCH C- algorithm:

$\{ \updownarrow(w0); \uparrow(r0, w1); \uparrow(r1, w0); \downarrow(r0, w1); \downarrow(r1, w0); \updownarrow(r0) \}$



fault	Condition	Sensitization	detection	Comments.
$\langle w_0/\downarrow\rangle$	a-cell < V-cell	M2	M2	N/A
$\langle w_0/\uparrow\rangle$	a-cell < V-cell	M4	M5	N/A
$\langle w_1/\downarrow\rangle$	a-cell < V-cell	M3	M4	N/A
$\langle v_1/\uparrow\rangle$	a-cell < V-Sell	M1	M1	N/A
$\langle r_0/\downarrow\rangle$	a-cell < v-cell	M3	M4	N/A
$\langle r_0/\uparrow\rangle$	a-cell < v-cell	M1	M1	N/A
$\langle r_1/\downarrow\rangle$	a-cell < v-cell	M2	M2	N/A
$\langle r_1/\uparrow\rangle$	a-cell < V-cell	M4	M5	N/A

fault	Condition	Sensitization	detection	Comments.
$\langle w_0/\downarrow\rangle$	v-cell < -cell	M4	M4	N/A
$\langle w_0/\uparrow\rangle$	v-cell < a-cell	M2	M3	N/A
$\langle w_1/\downarrow\rangle$	v-cell < a-cell	M1	M2	N/A
$\langle v_1/\uparrow\rangle$	v-cell < a-Sell	M3	M3	N/A
$\langle r_0/\downarrow\rangle$	v-cell < a-cell	M1	M2	N/A
$\langle r_0/\uparrow\rangle$	v-cell < a-cell	M3	M3	N/A
$\langle r_1/\downarrow\rangle$	v-cell < a-cell	M4	M4	N/A
$\langle r_1/\uparrow\rangle$	v-cell < a-cell	M2	M3	N/A

(ii) To modify MATS+ algorithm we consider all the patterns possible except all zero's and all one's.

the we write all '1' to each word in memory followed by all '0' for each word in memory.

for each bit we write pattern with only one '1' and rest of them to be '0' and one '0' with rest of them to be '1'

for each word we read and verify the data.

for B=4.

write the following patterns.

(i) 0001
(ii) 0010
(iii) 0100
(iv) 1000

then we write all '1' and all '0' pattern.

writing one '1' to each.

⇒ bit 0 → 1000
bit 1 → 0100
bit 2 → 0010
bit 3 → 0001

next we write one '0' to each.

⇒ bit 0 → 0111
bit 1 → 1011
bit 2 → 1101
bit 3 → 1110

then we read the data and verify.

(c) the restrictions to the march tests appear when we are trying to read or write to a specific memory address in the FIFO memory.
The other condition is reset where some tests like March C- doesn't work. For example March C- requires all the locations to be initialized to 0 in the start, but reset changes the values in the memory hence we cannot apply March C- test to FIFO.

Q4.
The paper "Memory Testing with a RISC Microcontroller" mainly discusses about testing low-end microcontroller which must apply low power on tests. They found out that the way to detect speed related faults is by performing memory tests at-speed, using Back-to-Back memory cycles which is minimum number of clock cycles for CPU. The paper uses ATMEL RISC microcontroller to demonstrate the capabilities and limitations of RISC CPU architectures.
It has 6 sections following the introduction which are in the following order:

2. ATMEL RISC microcontroller introduction
3. Explanation of the notation and specifying algorithms and shows test consequences of folding and scrambling.
4. Implementation of March elements.
5. A way to meet Back-to-Back cycle requirements.
6. Memory tests on ATMEL RISC CPU.
7. Conclusion.

Memory typically has a large number of words, while the number of bits per word is small. Folding is used such that the layout of the memory array approaches that of a square. It maps the logical memory view, consisting of words and bits/word, into the topological view, consisting of rows and columns. This paper assumes 4-way interleaved Folding.

Address scrambling means that the Logical address sequence, as applied from the outside of the memory, differs from the physical internal address sequence. A very common cause for address scrambling is via sharing in the address decoders.

This paper considers MARCH C- test with following algorithm:
{⇳(w0);⇑(r0,w1);⇑(r1,w0);⇓(r0,w1);⇓(r1,w0);⇳(r0)}
If the architecture also allows access to smaller data types, then, for speed-related faults in the data multiplexers the following tests are applied for each of the smaller data types:
SCAN+ {⇑(w0);⇑(r0); ⇑(w1);⇑(r1);⇓(w0); ⇓(r0);⇓(w1);⇓(r1)}
The following are the assumptions and conventions made:
4-way interleaved folding (F=4). Products may even have 16-way folding; this is a simple extension of F=4. No address and/or data scrambling is assumed (to keep the algorithms more comprehensible).

There are two key issues for Back-to-Back cycle requirements. They are 1. Loop counting and 2. Read result evaluation for 'r0' operations.

1. Loop counting:
   Consider the following:
   LDI R18, $M. Initialize Loop count
   CLR R31. Initialize Z-high
   LDI R30, $S. Initialize Z-low
   L1: ST Z+, R16. Perform a write operation

DEC R18. Decrement Loop count
BRNE L1. Branch if not equal
Here the problem is the memory was accessed only every 3 instructions in the loop which violates the Back-to-Back cycle requirement. So, the solution is Loop Unrolling which reduces the execution time by 58.3%. Similarly, paper addresses the case 2 problem also with example and solution which reduces the execution time by 37.5%.

Next the paper describes the capabilities and problems of ATMEL RISC architecture to implement memory test under Back-to-Back memory cycle requirements. It explains the testing in 2 sections first for data memory and then the program memory.
The following are the tests shown for data memory with examples:

1. MATS+ with linear Counting Method, Fast-Column and solid Data Background
2. MATS+ with linear Counting Method, Fast-Column and checkerboard Data Background
3. Address complement Counting Method
4. Fast-row Implementation.

Both the “MATS+ with linear Counting Method, Fast-Column and solid Data Background” and “MATS+ with linear Counting Method, Fast-Column and checkerboard Data Background” are explained with examples.
Address complement Counting Method is explained and proved with an example. It satisfies the Back-to-Back requirements using Z with post increment and Y with pre decrement Addressing Modes.
Fast-row addressing is not supported as it does not meet the Back-to-Back cycle requirement.
For program memory, both Fast-row and Address complement Counting Method cannot be applied, and it is concluded that Back-to-Back cycle requirements are not met.

Finally, the conclusion is that paper describes capabilities and faults of CPU-based memory testing. The tests in the paper are implemented in assemble language to satisfy Back-to-Back cycle requirements. It summarizes the data memory testing of ATMEL RISC CPU and then the program memory.

It concludes that tests must include the March C- to detest the static faults which are the state Coupling Faults (CFsts) and the static Address decoder Faults (AFs) can be detected with tests applied at any speed. it is recommended to provide more adequate architecture support for test implementation in next generation products. The results are a significant increase in fault coverage and a reduction in test time by about 60% when the test methods described in the paper are applied.